VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza a implementace požadavků firmy Gates Hydraulics s.r.o. pro podpůrné aplikace

Analysis and Implementatation of Requirements from Gates Hydraulics s.r.o. Company for External Applications

2014 Bc. Lubomír Fischer

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Lubomír Fischer**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Analýza a implementace požadavků firmy Gates Hydraulics s.r.o. pro podpůrné aplikace
Analysis and Implementation of Requirements from Gates Hydraulics s.r.o. Company for External Applications

Zásady pro vypracování:

Student se bude spolu s dalšími kolegy podílet na tvorbě IS pro firmu Gates Hydraulics, jež má sloužit pro správu a evidenci skladu a dalších externích procesů. Celý IS je poměrně rozsáhlý, proto se student zaměří hlavně na práci s prezentační vrstvou. Úkolem studenta bude vytvoření jednotného GUI celého IS s ohledem na jednoduchost a přehlednost. Funkcionalita GUI bude tvořena dle požadavků firmy.

Zadání práce lze shrnout do následujících bodů:
1. Seznámení se s problematikou fungování procesů ve firmě.
2. Získání požadavků firmy na nový informační systém.
3. Praktická realizace dle získaných požadavků, implementace prezentační vrstvy postavené na MVC a přizpůsobení zobrazení IS na malé displeje přenosných zařízení.

IS bude postaven na platformě .NET a očekává se jeho reálné uvedení do provozu.

Seznam doporučené odborné literatury:

[1] J. Chadwick, T. Snyder, H. Panda: Programming ASP.NET MVC 4, O'Reilly Media, 2012
[2] J. Tidwell: Designing Interfaces - Patterns for Effective Interaction Design, O'Reilly Media, 2009

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Václav Svatoň**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Student: Bc. Lubomír Fisher

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

Zveřejněna bude pouze veřejná část práce, která bude poskytnuta dle výše uvedených požadavků. Neveřejná část práce bude k dispozici pouze oponentovi práce a komisi pro potřeby obhajoby práce. Po obhajobě bude neveřejná část práce studentovi vrácena.

Student bude s neveřejnou částí práce nakládat v souladu s prohlášením v Rámcové dohodě o mlčenlivosti

V Karviné 2. května 2014

Ing. Anton Svrček, Gates Hydraulics s.r.o.

Rád bych na tomto místě poděkoval panu Ing. Václavovi Svatoňovi za cenné rady při psaní a také za vedení mé diplomové práce. Pak bych také rád poděkoval panu Antonu Svrčkovi za jeho poskytnutý čas při analýze a konzultaci celé vyvíjené aplikace. Také děkuji společnosti Gates Hydraulics s.r.o. za možnost vzniku této diplomové práce. V neposlední řadě bych chtěl zmínit poděkování panu Ing. Janu Martinovičovi, Ph.D. za odbornou konzultaci při řešení problematiky firmy Gates.

Abstrakt

Tato diplomová práce popisuje vyvíjenou webovou aplikaci pro firmu Gates Hydraulics s.r.o., která působí v rámci společnosti Gates Corporation. Webová aplikace poskytuje funkce, které jsou nutné k pracovním procesům ve firmě. Práce dále pojednává o celkovém vývoji této aplikace a popisuje jednotlivé kroky při vývoji a to od popisu pracovních procesů ve firmě,celkovou analýzu , tak až po samotnou implementaci aplikace. Tyto části jsou pak rozděleny do jednotlivých kapitol.

Klíčová slova: .Net, C#, Entity framework, MVC, Databáze, CSS, HTML, MS Visual Studio 2012, Diplomová práce

Abstract

This thesis describes the web application being developed for the company Gates Hydraulics Ltd., which operates within the Gates Corporation. The web application provides features that are required for work processes in the company. The thesis also discusses the overall development of the application and describes the various steps in the development and description of business processes in the company, the overall analysis, as to the actual implementation of the application. These parts are then divided into individual chapters.

Keywords: .Net, C#, Entity framework, MVC, Databases, CSS, HTML, MS Visual Studio 2012, Magister project

Seznam použitých zkratek a symbolů

ASP	–	Active Server Pages
BPCS	–	Business Planning and Control System
CSS	–	Cascading Style Sheets
GUI	–	Graphics User Interface
HTML	–	Hyper Text Markup Language
MS	–	Microsoft
MVC	–	Model View Controller
SO	–	Shoporder

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Společnost *Gates Hydraulics s.r.o.* působí v rámci společnosti *Gates Corporation* v Moravskoslezském regionu od roku 2005, kdy si od města Karviná pronajala výrobní halu v průmyslové zóně Nové Pole. Dnes společnost sídlí v nově postavené továrně, jejíž specializací je výroba kovových součástek pro hydraulické hadice a montážní celky pro evropský trh v oblasti stavební a zemědělské techniky.[1]

Cílem diplomové práce je vytvořit informační systém, běžící na dotykových zařízeních, pro sledování a plnění zásilek potřebnými součástkami a jejich celkový přehled při přijetí či plnění. Systém je vytvořen pro ulehčení práce zaměstnancům firmy a také pro celkový přehled výroby a dodávky. Diplomová práce obsahuje celkový proces při vyvíjení systému a to od dokumentu analýzy, který obsahuje vizi, analýzu, návrh, implementaci, tak po výstup, kterým je pak samotný informační systém.

Požadavek na vytvoření nového systému přišel ze strany firmy Gates z toho důvodu, že jejich nynější systém pracuje na již zastaralé technologii a také proto, že od doby, kdy systém byl vytvořen, vznikly úplně nové požadavky na systém a tudíž systém musel být neustále upravován. Z toho důvodu bylo rozhodnuto o požadavek na vytvoření nového systému, který bude pracovat nad databází jiného jejich novějšího systému *TrackMe* a to s tím, že se daná databáze rozšíří o chybějící tabulky z aplikace *Material Tracking System*.

V první části práce jsou popsány použité technologie, které jsou využity v samotné aplikaci. Následně je zobrazen popis toho, jak probíhají jednotlivé pracovní procesy ve firmě, kdy po jejich popsání je práce zaměřena již přímo na analýzu samotné webové aplikace. Předposlední část se zaměřuje na popis implementace a následně hlavní částí diplomové práce prezentační vrstvy. Ve finální fázi jsou shrnuty dosažené znalosti a ponaučení z tohoto diplomového projektu.

1.1 Cíl mé práce

Již při zadání vytvoření aplikace bylo vidět, že bude velice obsáhlá a pro jednoho studenta k vypracování příliš obtížná. Z toho důvodu byla aplikace rozvržena na dvě části a to na práci na datové vrstvě a práci na prezentační vrstvě. Pro tuto spolupráci byl pak i vytvořen *Team Foundation Server*[2] a využíván *Team Viewer*[3] pro co možná nejužší spolupráci na projektu.

Hlavním cílem této diplomové práce je popsat průběh vytváření prezentační vrstvy , jejího navržení a komunikací s vrstvou datovou. Obecně řečeno se jedná o přiřazení práce, v níž je potřeba navrhnout jednotlivé funkce systému a grafické rozhraní (GUI) s využitím datové vrstvy, kterou navrhoval kolega. V rámci grafického rozhraní se pak vše odráží od struktury prvků současného systému a to z toho důvodu, aby změna systému nebyla pro zaměstnance nepříjemná, ale aby byla pouze graficky přijatelnější a strukturově stejná, až na některé výhradní změny.

Práce má také druhou část a to je navržení grafického rozhraní pro mobilní zařízení, které slouží skladníkům pro skenování kódů z bedny a zobrazení skladů a jejich stavů. Této části je pak věnován až závěr implementační části práce, kdy se využívá v tu dobu

již navržené *GUI* pro webovou aplikaci. Grafický styl je na obou aplikacích aplikovaný vzhledově stejný, jen s pár odlišnými prvky.

2 Použité technologie

2.1 MVC

Základní myšlenkou *MVC*[4] architektury je oddělení logiky od výstupu. Architektura *MVC* se dělí na 3 logické celky. Těmi celky jsou *Model*, *View*, *Controller*. Každý z těchto celků plní svou úlohu v této architektuře.

Model je celek, který má na starost data a business logiku aplikace. V samotném modelu se pak definuje čtení a zápis do databáze. Funguje v podstatě na typu přijetí dat zvenku a výstupem dat ven. Samotný model pak obsahuje metody k výběru potřebných údajů z databáze.

View se stará o zobrazení uživatelského rozhraní aplikace uživateli a využívá k tomu data, která jsou vytvořena v modelu. V podstatě se jedná o již jednotlivé *HTML*[5] stránky. Přijímá data, která ale netuší odkud jsou a rovnou je předává na výstup, kde je zobrazuje uživateli.

Controller se pak stará o aplikační logiku a tok dat mezi vrstvami, aneb je to komponenta, která se stará o komunikaci s uživatelem, kdy předá data modelu a ten je předá databázi. Funguje jako komunikační prvek mezi uživatelem, modelem a view.

Výhodou samotného *MVC* je pak oddělení vrstev aplikace a jednodušší testování oproti jiným technologiím, například oproti *WebForms*[6]. Výhodou je pak také nezávislost jednotlivých komponent.

2.2 Entity framework

Jedná se o objektově-relační mapování. Slouží vytvoření všech tříd a vazeb z databáze, přímo do aplikace. Usnadňuje tak vytváření samotného datového modelu v aplikaci. Hlavní výhodou je, že klientská aplikace a databáze se mohou vyvíjet nezávisle na sobě.[7]

2.3 CSS

CSS[8] jinak řečeno kaskádové styly, slouží k nastylování vzhledu a umístění prvků v aplikaci. V dnešní době existuje již *CSS* verze 3, která obsahuje širokou škálu možností stylování vzhledu webových prvků, avšak né všechny prohlížeče podporují všechny tyto vlastnosti. Je také nezbytnou součástí při vytváření *GUI* aplikace.

2.4 HTML

Je jedním z hlavních jazyků pro vytváření webových stránek, který umožňuje publikace webových stránek na internetu. Samotná struktura *HTML* stránky je pak definována pomocí *tagů*, které obsahují jednotlivé prvky stránky, jakými jsou text, nadpisy, podnadpisy, tabulky, obrázky atd. Jednotlivé tagy se pak také dají označit pomocí tříd a jedinečných *ID*, které se následně mohou nastylovat pomocí kaskádových stylů. Poslední verze jazyka *HTML* je v současnosti verze 5. [9]

2.5 C Sharp

Vysokoúrovňový objektově orientovaný jazyk, který vyvinula firma *Microsoft*[10] spolu s platformou *.Net*[11]. Tento programovací jazyk je velice silný a lze s ním naprogramovat velkou řadu věcí, jakými jsou například desktopové aplikace, webové aplikace, formulářové aplikace a mnoho dalších. Neexistuje zde vícenásobná dědičnost, tudíž pak každá třída může být potomkem pouze jedné třídy.[12, 13]

2.6 Javascript

Interpretovací skriptovací jazyk, který běží na straně klienta, s podporou jednoduchého objektově orientovaného programování. Výhodou tohoto programovacího jazyka je jeho rychlost a také to, že nezatěžuje server, což je dáno tím, že běží právě na straně klienta. Je často vkládán do hlavičky *HTML* stránek.[14]

2.7 .NET Framework

Je základní komponentou platformy *.NET*, která zastřešuje všechny různé technologie softwarových produktů. Jedná se o prostředí, které je potřebné pro běh aplikací a nabízející jak spouštěcí rozhraní, tak i potřebné knihovny. Platforma *.NET* nepředepisuje konkrétní programovací jazyk, který se musí použít a tak je tedy možné provádět implementaci aplikací například v jazycích *C#*, *Visual Basic*, *Delphi* a dalších. Toto je zajištěno tím, jelikož se kód překládá do společného mezijazyka *Common Intermediate Language*.[11]

2.7.1 ASP.NET

ASP.NET[15] je součást *.NET Frameworku*, která je určena především pro tvorbu webových aplikací. Aplikace založené na této technologie jsou předkompilovány do *DLL* souboru, čímž pak může být způsoben rychlejší běh těchto aplikací, ve srovnání s ostatními skriptovacími jazyky. K dispozici je také *ASP.NET MVC Framework*, který poskytuje možnost vyvíjet aplikaci dle architektury *MVC* a nezávislost na *Javascriptu* oproti *WebForms*.[15]

3 Průběh zpracování zakázek ve firmě

Webová aplikace slouží firmě k celkové výrobě, skladování a dodávek produktů, které firma vyrábí. Celkový proces výroby se skládá z několika částí a kroků, které na sebe navazují ve výrobním procesu.

3.1 Výroba

Celý tento proces byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

3.2 Plnění

Celý tento proces byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

3.3 Odeslání (Reklamace)

Celý tento proces byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

3.4 Příjem

Celý tento proces byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

3.5 Vyjímečné případy

Celý tento proces byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

4 Analýza

Tato část diplomové práce popisuje analýzu celé webové aplikace a jednotlivé kroky, které následně slouží pro implementaci webové aplikace. V této části se pak nacházejí požadavky samotné firmy na systém a také popis důležitých kritérií pro funkčnost systému.

4.1 Vize

Základní vizí je naprogramovat webovou aplikaci, která bude sloužit jako pomoc při výrobě a dodávání součástek do zemědělských a stavebních strojů. Aplikace byla zažádána k vytvoření z toho důvodu, jelikož současná aplikace, kterou firma využívá, má různá omezení a zastaralou technologii. Bylo tedy rozhodnuto o vytvoření aplikace nové a sjednocení pod jejich hlavní webovou aplikaci *TrackMe*. Tato webová aplikace pak měla za úkol provádět jednotlivé pracovní procesy ve firmě, zobrazovat informace o skladě a různých pohybech daných beden a zakázek.

Aplikace je navržena tak, aby plnila veškeré funkce, které plnila jejich stávající aplikace s tím rozdílem, že je rozšířena o funkce další, postavena na novější technologii a také zohledněna o prvky, které je třeba vylepšit.

4.2 Role

Na základě konzultace a různých analýz současného stavu ve firmě, jsou určeny role zaměstnanců, kteří se systémem pracují, a práva jim určené, které jim povolují či omezují provádět jednotlivé procesy v aplikaci. Jednotlivé role jsou pak rozděleny následovně.

- **Administrátor** - uživatel přihlášený jako administrátor, může libovolně manipulovat s aplikací a její databází. Administrátor může také přidávat či odebírat jednotlivé zaměstnance.
- **Zaměstnanec** - uživatel přihlášený pod touto rolí má již omezený přístup do aplikace a k tomu se mu také vztahují jednotlivé procesy v aplikaci, které může provádět. Mezi procesy jemu povolené patří všechny typy naplnění beden, odeslání beden, kontrola přijatých beden a také zaslání beden na reklamaci či na pokovovací technologii.
- **Skladník** - toto je typ role, která se stará o přehled beden na skladu. S touto rolí může pak uživatel nahlížet do přijatých beden, spravovat zaměněné bedny a nahlížet do aktuálního stavu lokací všech beden nacházejících se u jednotlivých plejtařů.
- **Mistr směny** - tento typ uživatele má stejná práva jako skladník, avšak k nim má ještě možnost přistupovat k procesům aplikace jako jsou naplnění beden, odeslání beden a také přehled o dosavadních výkonech ostatních zaměstnanců.
- **Brigádník** - takto přiřazená role uživateli má omezená práva, ke kterým patří přístup pouze k přijetí a naplnění beden.

4.3 Požadavky na systém

Požadavky na systém jsou definovány samotnou firmou. Celkově se jedná o to, aby aplikace prováděla jednotlivé pracovní procesy ve firmě, které uměla již stávající aplikace, avšak s tím rozdílem, že bude stavěna na nové databázi, s rozsáhlejšími funkcemi a postavena na architektuře MVC. Toto všechno by pak mělo být navrženo tak, aby samotná aplikace prováděla jednotlivé akce rychleji a bezpečněji než předchozí aplikace.

Samotnou firmou je pak zadáno, s jakými problémy se u nynější aplikace potýkají a co konkrétně potřebují zlepšit. Mezi tyto požadavky patří například zvětšení dotykových prvků. Tento požadavek je určen z toho důvodu, aby byla usnadněna práce se systémem zaměstnancům, kteří jsou nuceni pracovat v pracovních rukavicích a tudíž mají stíženou práci s dotykovými prvky aplikace.

4.3.1 Funkční požadavky

Tato část diplomové práce popisuje funkční požadavky, které vzešly po mnoha konzultacích a různých jednání s daným programátorem firmy panem Antonem Svrčkem. Funkční požadavky jsou teoreticky definovány, jako požadavky na vyvíjenou webovou aplikace, ze kterých by pak měly vzejít funkce, které by systém měl umět. Stručně řečeno to jsou funkce, které má aplikace umět a splňovat.

V této sekci diplomové práce je vypsáno pouze pár hlavních funkčních požadavků. Všechny tyto funkční požadavky jsou dále popsány v dokumentu analýzy, který je přiložen jako příloha.

Mez hlavní funkční požadavky patří : Tyto požadavky byly označeny za neveřejné a nachází se tedy pouze v kompletní verzi diplomové práce.

4.3.2 Nefunkční požadavky

Mezi nefunkční požadavky se řadí takové požadavky, které omezují systém ve specifikovaných požadavcích na rámec vyvíjené platformy,architektury, výkonnosti a jiných dalších zohlednění aplikace, které nijak nesouvisí s jednotlivými funkcemi aplikace. Tato část diplomové práce popisuje jen pár vybraných nefunkčních požadavků. Zbytek nefunkčních požadavků lze nalézt v přikládaném dokumentu analýzy.

Nefunkční požadavky na aplikaci :

- **Rychlá odezva systému** - tento nefunkční požadavek je pro firmu velice důležitý, jelikož je jejich výrobní proces časově náročný a s každým ztraceným časem jim unikají finance. Proto je potřeba zohlednit to, aby jednotlivé procesy příliš dlouho netrvaly a aby reakce systému byla rychlá.
- **Webová aplikace bude vyvíjena na technologii MVC** - tento požadavek byl navržen firmě s tím, že aplikace bude vytvořena na této technologii, která přináší plno zvýhodnění oproti technologii *WebForms*. Celý popis této technologie MVC lze nalézt v 2 , kde je o ní napsáno více a jsou tam také rozebrány výhody i nevýhody.

- **Zohlednění pro interakci uživatele s pracovními rukavicemi** - tento požadavek patřil k jedním z prvotních požadavků, které firma poskytla. Řeší problém na pracovišti, kdy zaměstnanec je nucen z bezpečnostních důvodů pracovat s rukavicemi a tak má pak stíženou práci s dotykovým systém, kdy prvky aplikace jsou pro něj příliš malé. Toto zohlednění bylo bráno zřetel při vytváření *GUI* aplikace.
- **Webová aplikace bude mít přívětivé uživatelské rozhraní** - zde je řešen návrh GUI aplikace na základě struktury dosavadní aplikace, kde ovšem bude změněna celá grafika GUI tak, aby byly zohledněny všechny věkové kategorie zaměstnanců. Také je zde potřeba zohlednit toto rozhraní vůči ostatním uživatelům a tak je potřeba konzultace přímo s uživatelem, který se systémem pracuje.

4.4 Případy užití

Veškeré případy užití jsou definovány v dokumentu analýzy, který je námi zpracován ve spolupráci s Barborou Urbánovou, která se stará o vypracování všech těchto možných případů, které nastanou. Samozřejmě jednotlivé případy užití jsou výsledkem vzájemné spolupráce a konzultace nad danými problémy.

Plnění bedny - Celý tento případ užití byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5 Implementace webové aplikace

Implementace jednotlivých části je realizována na základě celého sestaveného dokumentu analýzy. Z dokumentu jsou postupně vybrány jednotlivé případy užití, které se následně realizují do samotné aplikace.

Aplikace je implementována na technologii *MVC* a *.Net frameworku 4.0*, jak již bylo zmíněno výše. Pro komunikaci s datovou vrstvu je vytvořena nadstavba *TrackMe.DataAccess*, která obsahuje veškerou komunikaci s databází a jednotlivé procesy jako je načtení procedur, funkcí a různé dotazy na databázi. Z této nadstavby jsou pak volány tyto funkce a procesy, prostřednictvím *controllerů* v hlavní aplikaci, které výsledné informace předávají dále modelům do samotných webových stránek a ve finální podobě je zobrazují uživateli.

Ve webové aplikaci je zohledněno umístění určitých prvků, které je potřeba zobrazit při každém kroku určitého procesu tak, aby se pozice a zobrazení těchto prvků neměnilo. Z toho důvodu bylo potřeba vytvořit pro tyto procesy jednu hlavní stranu se stálým zobrazení prvků, které pak načítá pracovní proces v každé fázi. Stránkám s těmito prvky se říká layouty a fungují obdobně jako *master.page*.

Pro celou aplikaci je také vytvořena jedna hlavní *master page*, která je nastylována pomocí *CSS* stylů. Samotná *master page* v sobě ukrývá hlavičku, která obsahuje nadpis s názvem prováděné akce, ve které se aplikace u vybraného procesu právě nachází.

5.1 Software a hardware

Pro vyvíjenou aplikací je také nutno brát v potaz software, na kterém aplikace běží. Firma má k dispozici webový prohlížeč *Internet Explorer* verze *10* a tudíž je aplikace naprogramována a nastylována tak, aby se vše korektně zobrazovalo a provádělo na této verzi prohlížeče.

Celá aplikace byla vyvíjena navržena v prostředí Microsoft *Visual Studio 2012*[16] a je součástí společného projektu. Samotná databáze je přizpůsobena pro plynulý chod na verzi *Microsoft SQL Server 2005*[17, 18].

Pro načítání beden firma využívá přenosné mobilní zařízení Motorola MC9090-G, které funguje jako čtečka kódu a provádí různé procesy, které jsou naprogramovány v mobilní aplikaci. Pro toto zařízení je brán v potaz rozměr displaye, jeho rozlišení a počet barev. Toto zařízení lze vidět na obrázku 1.

Obrázek 1: Přenosné mobilní zařízení
Obrázek převzat ze stránek [19]

5.2 Prezentační vrstva

Hlavním cílem diplomové práce je právě práce na této vrstvě aplikace. Prezentační vrstva slouží k zobrazení prvků aplikace uživatelům. V samotné prezentační vrstvě se pak nacházejí jednotlivé *HTML* stránky a jejich prvky, které se následně zobrazují uživateli. Prvky prezentační vrstvy, které se zobrazují uživateli do úhledné podoby se pak jako celek nazývají *GUI* aplikace, což je grafický vzhled celé aplikace.Tato vrstva také například kontroluje zadávané vstupy, avšak neobsahuje už jednotlivé zpracování dat.

GUI ve zkratce řečeno grafické uživatelské rozhraní. Podkladem k vytvoření *GUI* webové aplikace, je *GUI* současné aplikace firmy, ze kterého je zachována struktura rozložení prvků uživatelského rozhraní a to z toho důvodu, aby nebyl přestup ze staré aplikace na novou příliš složitý pro zaměstnance firmy a nemuseli tak hledat prvky aplikace jinde, než jsou zvyklí.

Samotné *GUI* aplikace je složeno z prvků, které jsou zobrazeny uživateli a také z grafického nastavení těchto prvků, které je provedeno pomocí *CSS* kaskádových stylů. V rámci této nové webové aplikace je pak celý *CSS* soubor vytvořen úplně nový a tak nezahrnuje žádné stylové prvky původní aplikace.

Níže lze vidět ukázku *CSS* stylu pro tlačítka ve webové aplikaci. Tlačítka jiných rozměrů jsou nastylována podobně, avšak s pár odlišnými vlastnostmi, jako je třeba zarovnání tlačítka, šířka, výška či jen posunutí od nějakého určitého prvku.

Příklad - CSS styl tlačítek

Ve výpisu 1, který lze najít pod tímto textem, jde vidět *CSS* styl pro samotné tlačítko, používané ve webové aplikaci. Tento styl je také použit pro více tlačítek v různých procesech aplikace.

- **height** - určuje výšku prvku u dané třídy.
- **min-width** - tato vlastnost nastavuje minimální šířku stylovaného prvku. Pokud by tedy prvek měl relativní šířku menší , než tuto minimální, tak se mu nastaví tato hodnota. Hodnota šířky může být určena v pixelech(px) nebo v procentech(
- **color** - vlastnost, která nastavuje barvu písma.
  - **!important** - pokud se k nějaké vlastnosti napíše tento výraz, pak danou vlastnost vynutí a dá přednost té vlastnosti, u které je přiřazen.
- **text-shadow** - vlastnost, která nastaví stín u textu. Jednotlivé hodnoty pak určují, kam bude stín směřovat. Skládá se ze tří hodnot a barvy. První hodnota nastavuje vržený stín po ose x, druhá hodnota nastavuje vržený stín po ose y a třetí hodnota nastavuje rozsah velikosti stínu.
- **border** - určuje velikost a barvu ohraničení u stylovaného prvku.
- **background** - touto vlastností se nastavuje danému prvku barva pozadí.
  - **linear-gradient** - toto je přidružená vlastnost u nastavovaného pozadí. Tímto lze nastavit prolínání barev od nějaké barvy po nějakou jinou barvu.

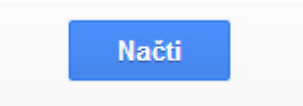
- **transition** - časová prodleva(přechod) při najetí myší na prvek. Lze určit jak dlouho bude trvat než se provede změna a také jaká změna se provede. Lze nastavit například zkosení, zmenšení, zvětšení, otočení prvku atd.
- **border-radius** - hodnota u této vlastnosti nastavuje míru zaoblení hran u daného prvku.
- **font** - touto vlastností se nastaví font písma, popřípadě jeho velikost a tloušťka.
- ***-webkit-*** - určuje výhradně vlastnost pro prohlížeč Google Chrome.
- ***-moz-*** - určuje výhradně vlastnost pro prohlížeč Mozzila Firefox.
- ***-ms-*** - určuje výhradně vlastnost pro prohlížeč Microsoft Internet Explorer.
- ***-o-*** - určuje výhradně vlastnost pro prohlížeč Opera.

```
.buttondetail , #btndetail, #btnErase, #btnOther
  {
        height: 30px;
          min-width: 80px;
        color: #FFF !important;
        text-shadow: 0 1px 0 #2F5BB7 !important;
        border: 1px solid #3079ED !important;
        background: #4B8DF8;
        background: -webkit-linear-gradient(top, #4C8FFD, #4787ED);
        background: -moz-linear-gradient(top, #4C8FFD, #4787ED);
        background: -ms-linear-gradient(top, #4C8FFD, #4787ED);
        background: -o-linear-gradient(top, #4C8FFD, #4787ED);
        background: linear-gradient(top, #4C8FFD, #4787ED);
        -webkit-transition: border .20s;
        -moz-transition: border .20s;
        -o-transition: border .20s;
        transition : border .20s;
        border-radius: 2px;
        -webkit-border-radius: 2px;
        -moz-border-radius: 2px;
        font: bold 13px Helvetica, Arial, sans-serif;

  }
```

Výpis 1: CSS styl tlačítka pro načtení

Níže na obrázku 2 můžeme vidět výsledné zobrazení tlačítka po aplikování popisovaného CSS stylu. Tímto tlačítkem se pak ve webové aplikaci načítají požadované informace v jednotlivých krocích procesu.



Obrázek 2: Tlačítko pro načtení prvků

Každé tlačítko na úvodní obrazovce zobrazuje jeden pracovní proces, který zaměstnanec provádí. Některé procesy však nebyly vůbec používány nebo nefungovaly a tak je toto zohledněno při vytváření nového *GUI*.

5.2.1 GUI a procesy

Celá tato část byla označena za neveřejnou a nachází se tedy pouze v kompletní verzi diplomové práce.

5.2.2 Views

Tato část struktury aplikace obsahuje jednotlivé *HTML* stránky, na kterých jsou následně zobrazeny jednotlivé prvky stránek uživateli. Jak je zmíněno výše, tak u některých procesů jsou vytvořeny layouty, které fungují na bázi *master page* a které zobrazují určité prvky v každé akci procesu.[3, 2]

Veškeré *Views* používají podobné části ve svém obsahu, jako je například definování modelu, předávání dat z modelu nebo volání controlleru, který náleží danému *View*. Jelikož je definování jednotlivých stránek podobné, tak jsou zde vybrány pouze dva případy těchto *Views* a popis toho, co který prvek znamená. Ostatní stránky lze vidět ve zdrojových kódech aplikace, kde se jich nachází daleko více. V podstatě lze říct, že každá stránka je jedno *View*, které má své vlastní prvky.

Pro ukázku se lze podívat na výpis 2 layoutu pro plnění beden, který má zajistit to, že tlačítka pro načtení, jinou zakázku a smazání, jsou neustále zobrazeny.

- **ViewBag.Title** - text u toho výrazu je následně zobrazen v hlavičce stránky a zobrazuje popis akce, která se právě provádí.
- **Layout** - cesta, která je nastavena u tohoto výrazu je cesta na chybovou stránku v případě, že nastane chyba. Pokud tedy nastane chyba, tak je zobrazena na stránce tato stránka s typem chyby, která snyní nastala.
- **@using (Html.BeginForm("LoadShopOrder", "InputBoxesSO"))** - tato část popisuje jaká akce nastane po stisknutí potvrzujícího tlačítka. V tomto případě když je stisknuto tlačítko, tak je zavolán *Controller InputBoxesSO* a v něm je následně zavolána akce s názvem *LoadShopOrder*.
  - **@Html.TextBox("shopOrderID")** - Tento výraz zobrazuje na stránce prvek pro zadání textu. Když je tento text vyplněn a potvrzen tlačítkem, tak je následně

uložen do proměnné *shopOrderID* a je volána akce *LoadShopOrder*. Tato akce požaduje na vstupu právě tento parametr shopOrderID, na základě kterého jsou načteny údaje do modelu a déle zobrazeny na stránce.

- **@RenderBody()** - tento příkaz slouží pro zobrazení těla následující stránky, která volá právě tento layout. Na tomto místě jsou pak zobrazeny prvky dané stránky.

```
@{
  ViewBag.Title = "Plneni beden";
  Layout = "~/Views/Shared/ActionDefaultPage.cshtml";
}

<div id="Searching">

    @using (Html.BeginForm("LoadShopOrder", "InputBoxesSO"))
    {
        <label class="label">Cislo S/O : </label> @Html.TextBox("shopOrderID")
        <input id="btndetail"  type="submit" value="Nacti" />
    }

    @using (Html.BeginForm("NewShopOrder", "InputBoxesSO"))
    {
        <input id="btnErase" type="submit" value="C" />
    }
    @using (Html.BeginForm("OtherShopOrder", "InputBoxesSO"))
    {
        <input id="btnOther" type="submit" value="Jina zakazka" />
    }

</div>

<div id="contentdetail">
  @RenderBody()
</div>
```

Výpis 2: View layout pro plnění beden

Načtení dat při plnění beden - pokud je tedy *layout* vytvořen i s požadovanými prvky, které je potřeba zobrazovat na každé stránce vybraného procesu, tak jej stačí použít jako cestu pro hodnotu *Layout* v úvodní stránce procesu, pro který se prvky mají zobrazovat

Na dalším výpise 3 níže je pak zobrazen další *layout*, který načítá prvky z layoutu výše. Navržené řešení je takhle vytvořeno z toho důvodu, jelikož ulehčuje práci a definování prvků na stránkách. Kdyby prvky, které se načítají na všech stránkách, nebyly takto řešeny, tak by data z modelu musely být předávány v každé stránce. Tímto řešením jsou přehledně definovány pouze na jedné stránce, ze které jsou načítány požadované informace k zobrazení.

- **@model** - tomuto výrazu se nastavuje cesta k modelu, ze kterého jsou zobrazeny načtené informace. Tento *model* je pak dále používán i pro další akce a je důležitý pro práci s daty na stránce.
- **if (@Html.ValidationSummary()** - tímto příkazem je na tomto místě zobrazována odchycená vyjímka a zároveň se tak ošetřuje na tomto místě a zobrazuje jí rovnou uživateli, aby věděl, co právě nastalo za chybu.
- **@if (Model != null)** - tato podmínka říká, že jestliže model není prázdný pak proveď námi požadovanou akci. V případě, který je níže, je určeno pod touto podmínkou to, pokud model není prázdný, tak zobraz požadované údaje z modelu.

```
@model TrackMe.DataAccess.Models.ShopOrder

@{
    ViewBag.Title = "_layout_input_shoporderform";
    Layout = "~/Views/Shared/InputBoxes/_LayoutPageInputBoxes.cshtml";
}

<div id="ValidationSummary">
   @Html.ValidationSummary()
  </div>

<div id="ListDetail">
  <h3>ShopOrder</h3>

  <table id="ListItem">
    <tr>
      <th>ShopOrder</th>
      <th>Item   </th>
      <th>Deskripce</th>
      <th>Pozadavek kusu</th>
      <th>StavSO</th>
      <th>Registrovanych kusu</th>
      <th>CelkovaVaha</th>

    </tr>

    @if (Model != null)
    {
      <tr>
        <td>@Model.ShopOrderID</td>
        <td>@Model.Material     </td>
        <td>Deskripce     </td>
        <td>@Model.PozadavekKusu</td>
        <td>@Model.State</td>
        <td>@Model.CelkemKusu</td>
        <td>@Model.CelkovaVaha</td>
      </tr>
    }
  </table>
</div>

  <div id="ListDetailR">
   @RenderBody()
  </div>
```

Výpis 3: View pro načtení beden

5.2.3 Mobilní aplikace

Mobilní aplikace slouží zaměstnancům při téměř každém pracovním procesu a proto je třeba ulehčit manipulaci s tímto zařízením a dát mu jednotný vzhled s nově vytvořenou webovou aplikací.

Ze všech konzultací s firmou pak ve výstupu vzešlo, že je potřeba pouze zohlednit GUI úvodní obrazovky s ohledem na uživatele, kteří musí pracovat s pracovním rukavicemi a tak jim je práce s mobilním zařízením stížená. Stížená je hlavně z toho důvodů, jelikož zařízení má poměrně malý display a prvky úvodní obrazovky současné mobilní aplikace mají špatné rozpoložení a tvar právě pro tuto manipulaci s rukavicemi.

Na základě těchto požadavků byla provedena analýza mobilního zařízení a tlačítka jsou nyní vytvořena tak, že jsou přehledná a lehce stisknutelná bez nechtěného překliknutí na jiný proces. Se současným vzhledem již nyní nedochází k problémům, kdy uživatel omylem stiskne vedlejší proces než ten, který vlastně chce. Toto také zrychluje pracovní proces, jelikož při každém překliku se ztrácí čas, kterým je uživatel nucen se vrátit na úvodní obrazovku. Výše zmiňovanou úvodní obrazovku mobilní aplikace lze vidět na obrázku 3.

Obrázek 3: Hlavní nabídka mobilního zařízení

5.3 Datová vrstva

Tato část diplomové práce spadá k vytvoření na mého kolegu Miroslava Zajonce, avšak byla tvořena současně s prezentační vrstvou a výsledná forma datové vrstvy je nakonec práce nás obou.

Základem pro datovou vrstvu je zaměření se na velice rozsáhlý databázový systém *BPCS*[20], na kterém firma staví všechny své databáze různých webových aplikací. Tento databázový systém, se stará o jednotlivé operace na pozadí, jako je uzavírka tabulek, generování *shoporderu* a dalších mnoha věcí, do kterých ani nelze pořádně vidět, jelikož rozsáhlost tohoto systému je opravdu veliká a ani není tak velká potřeba jej pochopit pro vyvíjenou webovou aplikaci až na některé databázové tabulky.

Celá navržená datová vrstva s databází je pak stavěna nad databází aplikace *TrackMe*, přes kterou firma kontroluje veškeré informace a provádí další pracovní procesy. Databáze této aplikace obsahuje již nějaké stejné tabulky jako aplikace předchozí, avšak řada z nich je nyní doplněna o změny.

Důvodem rozšíření této stávají databáze bylo to, že v budoucnu by mohly všechny nové aplikace firmy pracovat na jedné kompletní databází a být spravovány přes jednu hlavní aplikaci.

5.4 Chybové stavy

Tato část je pro firmu velice důležitá a také je definována po mnoha letech zkušenosti práce se systémem. Firma zažila řadu situací, při kterých nastalo znemožnění provedení některých pracovních procesů a u je u nich potřeba zásahu do systém.

Ve webové aplikaci proto je potřeba ošetřit mnoho nečekaných případů, které by mohly nastat. Mezi nečekané případy řadíme například stav, kdy uživatel zadává číslo bedny, která není ještě v databázi nebo bedna je v databázi ale je již naplněna. Obě tyto situace jsou řešeny jinak.

5.4.1 Zadání neexistující bedny

Celý tento popis chybového stavu byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5.4.2 Zadání již naplněné bedny

Celý tento popis chybového stavu byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5.4.3 Roztrhnutý štítek

Celý tento popis chybového stavu byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5.4.4 Jiná váha dílu

Celý tento popis chybového stavu byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5.4.5 Chybí tara bedny

Celý tento popis chybového stavu byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5.4.6 Chceme pokovit pokovený materiál

Celý tento popis chybového stavu byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5.4.7 Chceme přijmout již přijatou nebo neodeslanou bednu

Celý tento popis chybového stavu byl označen za neveřejný a nachází se tedy pouze v kompletní verzi diplomové práce.

5.5 Algoritmy

Algoritmus by se dal popsat jako předpis pro řešení konkrétní úlohy, kde se dané řešení skládá z kroků, u nichž je zabezpečeno to, že jsou na základě dat na vstupu poskytnuta data výstupní. Každý algoritmus pak musí splňovat následující vlastnosti :

- **Konečnost** - výsledek, který je očekáván na výstupu musí být proveden v rozumném čase. Tento čas je v ideálním případě ten, kdy výsledek k něčemu bude,tudíž by měl být proveden v co nejmenším čase.
- **Hromadnost** - vstupní data netřeba chápat jako konkrétní hodnoty na vstupu, ale spíše jako množinu dat na vstupu, ze kterých si lze vybrat.
- **Jednoznačnost** - algoritmus je složen z jednotlivých kroků, které na sebe navazují. Každý tento krok lze chápat jako přechod z jednoho stavu algoritmu do druhého. Každý stav je pak dán určitými daty a to tak, že tím jak data vypadají pak musí být jednoznačně určeno, jaký krok bude následovat.
- **Opakovatelnost** - při použití totožných dat na vstupu musí být obdržen pokaždé totožný výsledek.
- **Rezultativnost** - algoritmus musí vést ke správnému výsledku Při použití totožných dat na vstupu se musí pokaždé dostat stejný výsledek.

Příklad - porovnání položek
Celý tento algoritmus lze vidět na výpisu 4.

- Jako vstupní data lze u spočtení kusů chápat množinu dat, obsahující zadávanou váhu jednoho kusu, zadávanou celkovou váhu, porovnávanou váhu kusu a taru bedny. Tyto hodnoty na vstupu musí být zadávány, jelikož jsou požadovány pro výpočet počtu kusů v bedně.
- Jednotlivé kroky tohoto algoritmu je pak možnost popsat tak, že v první kroku se stanovuje podmínka toho typu, že jestliže převod zadávané váhy formátu řetězce string na typ *double* je prázdný, pak je vyvolána vytvořená vyjímka *WrongInputException* , která dostává jako paramtr string hlášku, že *"Zadaná celková váha 0 nebyla ve správném formátu"* , kde 0 nám zobrazuje zadanou váhu *inputWeightstr* . Pokud je celková váha uživatelem zadávána dobře, pak se váha převede na výstup *inputWeight* , který je typu *double*.
- V dalším kroku je stejným způsobem převáděna vkládaná váha jednoho kusu materiálu. Opět je převáděna váhu kusu na typ *double* z řetězce string. Pokud není

možnost převést, pak je vyvolána opět vyjímka *WrongInputException*, která dostává jako parametr string *"Zadaná váha kusu 0 nebyla ve správném formátu"* . Jestliže je však vložena váha kusu ve správném formátu, tak je váha kusu převedena do proměnné *inputItemWeight*, která je typu *double*.

- Následně je stanovena další podmínka pro vkládanou celkovou váhu. Jestliže je převedená celková váha menší nebo rovna nule, pak se vyvolává opět vyjímka, která dostane jako předaný parametr string *"Zadaná celková váha 0 musí být nezáporná a větší než 0"* .

- Tato stejná podmínka je pak stanovena i pro zadávanou váhu jednoho kusu, kde ale v případě, že zadávaná váha kusu je větší nebo rovna nule, je vyvolána stejná vyjímka, avšak s jiným parametrem string, který je *"Zadaná váha kusu 0 musí být nezáporná a větší než 0"* . Pokud uživatel nezadá všechny vstupní informace v požadovaném tvaru, tak mu vyvolaná vyjímka ukáže, co je špatně zadáno a tak je uživatel nucen vyplňované informace na vstupu opravit tak, aby vyhovovali požadovaným informacím ve správném tvaru. í nebo rovna nula,

- Těmito podmínkami je pak stanoveno to, že daný uživatel

- Při správném vyplnění všech informací lze pak vytvořit instance třídy *ItemComparisionResult* pod názvem *iCr* , která má dvě proměnné a těmi jsou *ItemCount* typu *int* a *Netto* typu *double*.

- Po vytvoření této instance je vytvořena nová nulová proměnná *binTara* typu *double*, do které je z databáze , pomocí procedury *GetBinTarra*, načtena váha bedny na základě zadaného kódu bedny. Pokud je *binTarra* nulová, pak je vyvolána vyjímka se string parametrem *"Doplňte TARRU prosím"*. Do proměnné *Netto* se následně ukládá celková váha kusů, bez váhy bedny. Ta je určena určena tím, že od celkové zadané váhy se odečte proměnná *binTarra*, která definuje váhu samotné bedny.

- Pokud je pak vkládaný typ *LotID* a nikoli materiál samotný, pak je do nově vytvořené proměnné *itemOriginialW* vkládána váha kusu, se kterým se má být porovnávána zadávaná váha a to pomocí metody *GetMaterialInfoByLotID*, která určuje váhu kusu na základě $lotId$. Pokud není typ *LotId*, tak je váha porovnávaného kusu určena na základě metody *GetMaterialWght*, které se předává typ materiálu a následně se z databázové tabulky materiál vytahuje váhu kusu na základě zadávaného materiálu.

- Dále se porovná váha nově naplněné proměnné *itemOriginialW* na nenulovost a pokud je váha nulová, vyvoláme vyjímku *EmpyResultException* s parametrem *"U materialu k lotu 0 nebyla zadana hmotnost"*.

- Ve finální fázi tohoto algoritmu je naplněna proměnná *ItemCount* třídy *ItemComparisionResult* , pomocí metody *GetItemCount*,které jsou předávány parametry *inputWeight*, *inputItemWeight*, *(double)itemOriginialW*, *(double)binTara*. Tato metoda následně spočte počet kusů materiálu v bedně na základě zadaných parametrů.

- Jako výstupem celého tohoto algoritmu je instance třídy *ItemComparisionResult*, která nese vyplněné proměnné *Netto* a *ItemCount*.

```
public ItemComparisionResult CompareItems(string inputWeightstr, string inputItemWeightstr,
    string binNo, string item, ItemType itemtype )
  {
    double inputWeight;
    if (!double.TryParse(inputWeightstr.Replace(" ", ""), out inputWeight))

      throw new Exceptions.WrongInputException("Zadana celkova vaha {0} nebyla ve
          spravnem formatu", inputWeightstr);


    double inputItemWeight;
    if (!double.TryParse(inputItemWeightstr.Replace(" ", ""), out inputItemWeight))
      throw new Exceptions.WrongInputException("Zadana vaha kusu {0} nebyla ve spravnem
          formatu", inputItemWeightstr);

    if (inputWeight < 0 || inputWeight == 0)
      throw new Exceptions.WrongInputException("Zadana celkova vaha {0} musa byt
          nezaporna a vetsi nez 0", inputWeight);

    if (inputItemWeight < 0 || inputItemWeight == 0)
      throw new Exceptions.WrongInputException("Zadana vaha kusu {0} musi byt nezaporna
          a vetsi nez 0", inputItemWeight);


    ItemComparisionResult iCr = new ItemComparisionResult();



    decimal? binTara = new BinHandler(Context).GetBinTarra(binNo);

    if (binTara == null || binTara == 0)
    {
      throw new Exceptions.BinTaraNotFound("DopInte TARRU prosim");
    }


    // zjistim  vahu obsahu
    iCr.Netto = inputWeight - (double)binTara;
    decimal itemOriginialW = 0;

    if (itemtype == ItemType.LOTID)
      itemOriginialW = GetMaterialInfoByLotID(item).Weight.GetValueOrDefault();
    else
      itemOriginialW = GetMaterialWght(item);

    if (itemOriginialW == (decimal)0.0)
    {
      throw new Exceptions.EmpyResultException("U materialu k lotu {0} nebyla zadana
          hmotnost", item);
    }

    iCr.ItemCount = GetItemCount(inputWeight, inputItemWeight, (double)itemOriginialW, (
        double)binTara);
    return iCr;
  }
```

Výpis 4: Ukázka porovnávání prvků na vstupu

Příklad - vypočtení kusů v bedně
Celou ukázkou tohoto algoritmu lze vidět na výpisu 5.

- Cílem tohoto algoritmu je zjištění počtu kusů materiálu v bedně. Vstupem tohoto algoritmu jsou jednotlivé váhy, které uživatel zadává při požadované operaci. Na vstupu je tedy požadováno zadaní celkové váhy, váhy jednoho kusu, porovnávanou váhu kusu a váhu prázdné bedny. Na výstupu se očekává číselná hodnota typu *int*.
- Při zavolání tohoto algoritmu je hned na začátku vytvořena proměnná netto typu decimal, která zde vyjadřuje celkovou váhu materiálu v bedně. Tato váha je určena tím, že od celkové zvážené váhy plné bedny se odečítá váha prázdné bedny.
- V dalším kroku se pak stanovuje podmínka, ve které je volána metoda *CompareItemWeight* . Tato metoda porovnává váhu zadaného kusu a váhu porovnávaného kusu, kde na svém výstupu vrací *true*, pokud je váha kusu shodná nebo je v povolené toleranci a *false*, pokud není shodná či se nevlezla do povolené tolerance.
- Pokud však podmínka vrátí booleovskou hodnotu *true*, pak se na výstupu vrací celkový počet kusů materiálů v bedně, na základě celkové zvážené váhy bedny, od které je odečítána váha prázdné bedny, kdy se tato výsledná hodnota, která je obdržena po daném odečtení, podělí váhou jednoho kusu materiálu v bedně.
- Pokud je podmínka false, pak je vyvolána vyjímka *ItemCoparisionException* a předáme ji *string* parametr *"Zadana vaha 0 u Itemu neodpovida jeho vahove toleranci. Prevazte dil!"*.

```
public int GetItemCount(double inputWeight, double inputItemWeight, double originalItemW,
    double tara)
{
  decimal netto = (decimal)(inputWeight − tara);

  if (CompareItemWeight((double)originalItemW, inputItemWeight))
  {
    return (int)Math.Round((inputWeight − tara) / originalItemW, 0);
  }
  else
    throw new Exceptions.ItemCoparisionException(Exceptions.ExceptionLevel.WARNING,
      "Zadana vaha {0} u Itemu neodpovida jeho vahove toleranci. Prevazte dil!",
          inputItemWeight);
}
```

Výpis 5: Počet kusů v bedně

5.6 Struktura hlavních prvků aplikace

TrackMe.Web

Celá webová aplikace je rozdělena do stromové struktury přehledně uspořádaných prvků. Na stromové struktuře níže lze vidět, jak je celá aplikace rozvrstvena a co které prvky znamenají. Je zde zobrazeno také rozvrstvení technologie MVC položky, které se nachází v různých části *models*, *views*, *controllers*.

- **Controllers** - Tato vrstva technologie MVC obsahuje jednotlivé metody a funkce, které pracují jak s databází, tak i s jinými prvky aplikace. Tyto metody ověřují údaje a následně na základě vstupních parametrů a parametrů v uložených procedurách zpracovávají data a následně je zobrazí na výstupu nebo uloží do určeného modelu.
  - AcceptanceBoxesController - *Controller pro přijetí pro proces přijetí beden*
  - ActualStateLocationController - *Controller pro přijetí pro proces zobrazení stavu lokací*
  - BaseController - *Controller komunikaci s DataAcess*
  - BinToShipmentController - *Controller pro přijetí přidávání beden na shipment*
  - ComplaintController - *Controller pro přijetí reklamace*
  - ErrorController - *Controller pro chybové stavy*
  - HomeController - *Controller pro přijetí pro úvodní menu*
  - ChangedBoxesController - *Controller pro přijetí pro proces zaměnění beden*
  - InputBoxesItemController - *Controller pro proces plnění beden bez SO*
  - InputBoxesQALController - *Controller pro proces plnění beden z reklamace*
  - InputBoxesSOController - *Controller pro proces přijetí beden*
  - LoginController - *Controller pro proces přihlášení uživatele*
  - MoveBoxesController - *Controller pro proces přesunutí beden*
  - PrintBarController - *Controller pro proces dotisknutí štítků*
  - ShipmentController - *Controller pro odeslání beden*
- **Extensions** - Rozšíření
  - BinExtension - *Rozšíření pro bedny*
  - ShipmentExtension - *Rozšíření pro shipment*
  - ShopOrderExtension - *Rozšíření pro shoporder*
- **Model** - Tato část aplikace popisuje modely, které se vyskytuje ve vrstvě model technologie MVC. Zde jsou určeny všechny data, která jsou načtena z databáze a která se následně zobrazují v HTML stránce.
  - Bins - *Model s údaji pro bedny*

  - Boxes - *Model s údaji pro bedny*
  - DatePickerModel - *Model s údaji ohledně generování času*
  - ErrorModel - *Model s údaji pro chybové stavy*
  - LoginModel - *Model pro přihlášení*
  - SentShipment - *Model pro odeslání beden*
  - Shipments - *Model pro shipmenty*

- **Views** - Slouží pro zobrazení jednotlivých HTML stránek, které čerpají data jak z modelu, tak z controlleru, kterému posílají údaje na základě odeslání akce, která pošle parametr na základě čeho se jí pak vrátí validní či nevalidní model.

  - AcceptanceBoxes - *Proces přijetí beden*
    - AcceptBin.cshtml - *Stránka pro přijetí bedny*
    - CheckBin.cshtml - *Zkontrolování údajů před přijetím*
    - Index.cshtml - *Úvodní stránka procesu*
    - LoadBoxes.cshtml - *Načtení bedny pomocí kódu bedny*
  - ActualStateLocation - *Proces aktuální stav lokací*
    - Index.cshtml - *Úvodní stránka s přehledem lokací*
  - Error - *Stránky při vyvolání chyby*
    - Index.cshtml - *Úvodní chybová stránka*
    - NotFound.cshtml - *Chybová stránka pro nenalezení prvku*
  - Home - *Úvodní stránka se všemi procesy*
    - Index.chtml - *Úvodní stránka aplikace*
  - ChangedBoxes - *Proces zaměněné bedny*
    - Index.cshtm - *Úvodní stránka procesu*
    - NexStep.cshtml - *Stránka dalším krokem pro záměnu*
    - Change.cshtml - *Stránka,která provede záměnu bedny*
  - InputBoxesItem - *Proces plnění beden bez SO(pomocí LotID)*
    - AcceptItem.cshtml - *Naplnění bedny*
    - CheckItem.cshtml - *Stránka s kontrolou zadaných údajů*
    - Index.cshtml - *Úvodní stránka procesu*
    - LoadItem.cshtml - *Stránka s načtenými údaji*
  - InputBoxesQAL - *Proces plnění beden z reklamace*
    - Index.cshtml - *Úvodní stránka procesu*
    - LoadQAL.cshtml - *Stránka s načtenými údaji*
    - CheckQAL.cshtml - *Stránka s kontrolou zadaných údajů*
    - AcceptQAL.chtml - *Naplnění bedny*

- InputBoxesSO - *Proces plnění beden pomocí SO*
  - FillBin - *Naplnění bedny*
  - CheckBin - *Stránka s kontrolou zadaných údajů*
  - Index - *Úvodní stránka procesu*
  - LoadShopOrder - *Stránka s načtenými údaji*
- Login - *Proces pro přihlášení/registraci uživatele*
  - CreateUser - *Stránka procesu vytvoření uživatele*
  - Index - *Úvodní stránka procesu*
  - LoggIn - *Stránka procesu přihlášení uživatele*
  - LogOut - *Stránka procesu odhlášení uživatele*
- MoveBoxes - *Proces přesunutí bedny*
  - Index - *Úvodní stránka procesu*
- PrintBar - *Proces dotisknutí štítků*
  - Index - *Úvodní stránka procesu*
  - Location - *Stránka zobratení beden dle lokace*
- Shared - *Složka se sdílenýma stránkama pro různé procesy*
  - AcceptanceBoxes - *Prvky pro přijetí beden*
    - $_layout_accept_boxesform.cshtml$ - *Layout pro příjem beden*
    - $_layoutPageAcceptanceBoxes.cshtml$ - *Layout pro příjem beden*
  - ActualStateLocation - *Prvky pro aktuální stav lokací*
    - $Layout_ActualStateLocation.cshtml$ - *Layout pro zobrazení aktuálního stavu lokací*
    - $Layout_ActualStateLocation_Base.cshtml$ - *Layout pro zobrazení aktuálního stavu lokací*
  - InputBoxes - *Prvky pro plnění beden*
    - $_layout_input_shoporderform.cshtml$ - *Layout pro plnění beden*
    - $_LayoutPageInputBoxes.cshtml$ - *Prvky pro plnění beden*
  - InputBoxesItem - *Prvky pro plnění beden bez SO*
    - $_layout_InputBoxesItem_base.cshtml$ - *Layout pro plnění beden bez SO*
    - $_layout_InputBoxesItem_table.cshtml$ - *Layout pro plnění beden bez SO*
  - PrintBar - *Prvky pro dotisknutí štítků*
    - $_layout_PrintBar.cshtml$ - *Layout pro dotisk štítků*
    - $_layout_PrintBarForm$ - *Layout pro dotisk štítků*
  - ShipmentOverView *Přehled odeslaných beden*
    - $ShipmentOverView_Base$ - *Layout přehled odeslaných beden*
  - $_LayoutPage.cshtml$ *Hlavní layout stránka*
  - $_LayoutPage_SendBoxes.cshtml$ - *Stránka s prvky pro proces odeslání beden.*
  - ActionDefaultPage.cshtml - *Stránka pro kontrolu chyb*

        * ErrorPage.cshtml - *Chybová stránka*
        * Site.Master - *MasterPage*
    - Shipment
        * Detail - *Zobrazení podrobností shipmentu*
        * Index - *Úvodní stránka procesu*
        * NewShipment - *Založení nového shipmentu*
        * Warning - *Chybová stránka*

- **Content** - V této složce lze nalézt CSS styly pro jednotlivé prvky stránek, všechny obrázky využívané ve webové aplikaci a také různé knihovny jazyka *javascript*.
    - themes - *Složka s obrázky pro webové stránky*
    - Style1.css - *Soubor s kaskádovými styly pro prvky webových stránek*

TrackMe.DataAccess - Tento projekt slouží jak nadstavba pro celou webovou aplikaci. Jsou v ní uloženy handlery i další modely, které jsou potřeba pro práci s daty. Jsou zde také definovány vlastní vyjímky a celá komunikace s databází.

- **Exceptions**
    - BinHandleException *Stránka s jednotlivým metodami pro operace s bednami*
    - EmpyResultException *Stránka s vlastními vyjímkami*
    - ExceptionLevel *Úroveň vyjímky*
    - MaterialException *Chybové stavy pro materiál*
    - PlatingExceptions *Chybové stavy pro plejtaře*
    - SameShipmentExistException *Chybové stavy pro shipmenty*
    - ShipmentOperationException *Chybové stavy pro operaci se shipmentem*
    - TrackMeException *Vlastní vyjímky*
    - WrongInputException *Vyjímky pro špatné zadávání*

- **Handlers**
    - BinHandler *Stránka s metodami pro operaci s bednami*
    - LoginHandler *Stránka s metodami pro přihlášení a odhlášení*
    - MaterialHandler *Metody a funkce pro práci s materiálem*
    - ShipmentBinHandler *Metody a funkce pro práci s shipmentem a bednou*
    - ShipmentHandler *Metody a funkce pro práci s shipmentem*
    - ShipmentItemHandler *Metody a funkce pro práci s položkou v shipmentu*
    - ShopOrderHandler *Metody a funkce pro práci s shoporderem*

- **Models**
  - Bin *Model pro bednu*
  - BinControlInput *Model pro ověření zadávaných údajů pro bednu*
  - IModel *Model pro kontrolu*
  - ItemComparisionResult *Model pro porovnání dvou prvků*
  - LotItem *Model pro LotID*
  - MyDate *Model pro generování datumu*
  - Plater *Model pro plejtaře*
  - Shipment *Model pro shipment*
  - ShipmentItem *Model pro položku v shipmentu*
  - ShopOrder *Model pro shoporder*

6 Testování

Po úspěšné implementace celé webové aplikace přichází samotné testování a ověření správnosti jednotlivých funkcí

Testování je proces, ve kterém se simulují jednotlivé případy, které mohou nastat při práci s aplikací. Celé toto testování je prováděno za účelem ověření správnosti funkcí a také k nalezení všech možných chyb, před nasazením aplikace do provozu.

Testovat můžeme rozdílnými způsoby, které se třídí na manuální testování a automatické testování. Oba typy testování mají své výhody i nevýhody. Určitě jedním z větších rozdílů je čas, který u testování strávíme a také to jak jsou testy prováděny.

Automatické testy - Tento typ testu je výhodné použít při opakovaném spouštění velkého množství testů, nebo testu, který obsahuje velké množství generovaných dat. Samotné provádění testů je pak ověřováno tak, zda je zjištěno nějaké nestandardní chování oproti testu předešlému. Pro použití automatizovaného testování je potřeba mít testovací nástroj, vstupní a výstupní data, která následně porovnávám a vyhodnocujeme a v poslední řadě je potřeba mít testovaný program.

Manuální testování - používá se bez jakýchkoliv testovacích nástrojů a skriptů. U manuální testování testujeme určité neočekávané události, které mohou nastat. U těchto testů tester provádí celý test.

Ve vyvíjené aplikaci byly využity automatické testy, s využitím jednotlivých unit testů a dále byly využity testy na straně uživatele. Testy na straně uživatele se nazývají Web Performance Test[21] a jsou součástí vývojářského prostředí Microsoft Visual Studio 2012. Tyto testy fungují na principu ověření správnosti zobrazení dat na základě procházení webu uživatelem.

Příklad testování výpočtu kusů - tento příklad 6 znázorňuje samotný unit test, který porovnává položky na vstupu a následně vyhodnocuje zda-li je na základě zadaných a načtených údajů totožně spočten počet kusů materiálu. Na začátku testu je třeba si vytvořit cestu k databázi a k handleru, který obsahuje potřebné metody k provedení daného testu. Test se skládá ze dvou fází, kdy v první fázi se definují parametry na vstup operace a očekávaná hodnota na výstupu. V druhé fázi se pak zavolá metod,která provádí výpočet počtu kusů na základě zadaných údajů. Celý test je završen tím, že se porovnává očekávaná hodnota s hodnotou na výstupu operace výpočtu kusů.

```
[TestMethod()]
        public void GetItemCountTest()
        {
            DbContext cn = new DbContext(@"Data Source=lubko\sqlexpress;Initial Catalog=
                TrackMePLT;Persist Security Info=True;User ID=sa;Password=6419900", "mz");
            MaterialHandler target = new MaterialHandler(cn);
            double inputWeight = 30;
            double inputItemWeight = 0.048;
            double originalItemW = 0.048;
            double tara = 1;
            int expected = 604;
            int actual;
            actual = target.GetItemCount(inputWeight, inputItemWeight, originalItemW, tara);
            Assert.AreEqual(expected, actual);

        }
```

Výpis 6: Testování výpočtu celkových kusů

Příklad testu načtení bedny - na tomto příkladu 7 lze vidět testování načtení samotné bedny na základě jejího čísla. Opět je důležité si nastavit cestu k databázi a samotnému handleru, který obsahuje metody k práci s bednou. V prvotní fázi se nastaví do proměnné *binNo* string parametr obsahující číslo bedny. V dalším kroku se vytvoří nový model bedny, do kterého se přiřadí tato nová hodnota čísla bedny. Následně se provede načtení bedny pomocí metody *GetBinByPrindt*, která dostane jako parametr na vstupu právě toto číslo bedny. Výsledek se stanovuje na základě toho, zda-li je počet nalezených beden shodný s tím, který se nachází v očekávaném modelu.

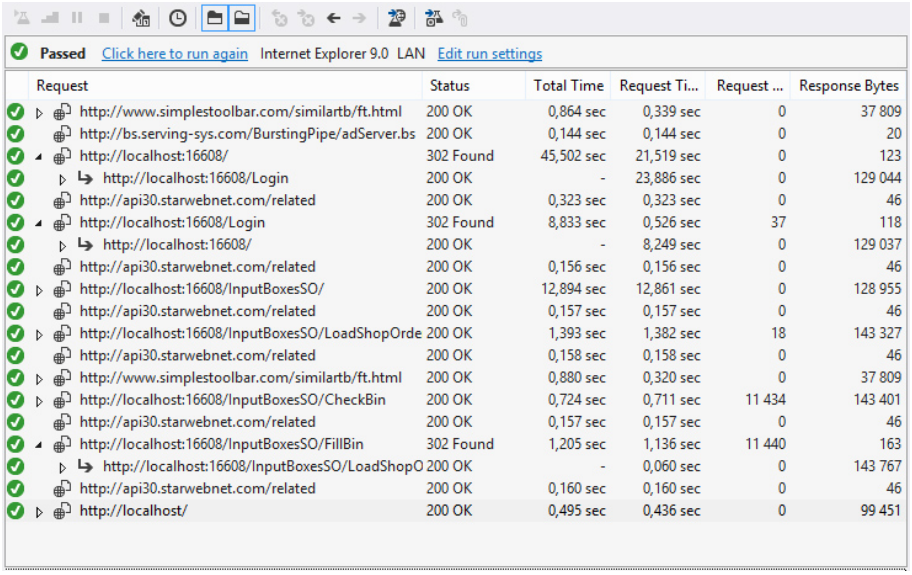
```
[TestMethod()]
        public void GetBinByBinPrintTest()
        {
            DbContext cn = new DbContext(@"Data Source=lubko\sqlexpress;Initial Catalog=
                TrackMePLT;Persist Security Info=True;User ID=sa;Password=6419900", "mz");
            BinHandler target = new BinHandler(cn);
            string binNo = "BI00201115K";
            Bins expectedBin = new Bins();
            Bin expected = new Bin();
            expected.binNo = binNo;
            expectedBin.Add(expected);
            Bins actual;
            actual = target.GetBinByBinPrint(binNo);
            Assert.AreEqual(expectedBin, actual);

        }
```

Výpis 7: Test načtení bedny

Příklad web performance test - tento příklad popisuje testování aplikace na straně uživatele. Celý tento test má za úkol otestovat celý proces naplnění beden. V testu lze

vidět kroky, kterými se prochází při proklikávání jednotlivých části procesu naplnění beden. Obrázek 4 zobrazuje podrobný výpis výsledků celého testu. Po jeho shlédnutí si lze všimnou časových údajů v kolonkách *total time* a *request time*. Kolonka *total time* je doba, za jakou trvalo načtení požadavků a všech jeho závislostí. Čas v kolonce *request time* zobrazuje celkovou dobu odezvy z webového serveru po dokončení individuální žádosti.



Passed Click here to run again Internet Explorer 9.0 LAN Edit run settings

Request	Status	Total Time	Request Ti...	Request ...	Response Bytes
http://www.simplestoolbar.com/similartb/ft.html	200 OK	0,864 sec	0,339 sec	0	37 809
http://bs.serving-sys.com/BurstingPipe/adServer.bs	200 OK	0,144 sec	0,144 sec	0	20
http://localhost:16608/	302 Found	45,502 sec	21,519 sec	0	123
http://localhost:16608/Login	200 OK	-	23,886 sec	0	129 044
http://api30.starwebnet.com/related	200 OK	0,323 sec	0,323 sec	0	46
http://localhost:16608/Login	302 Found	8,833 sec	0,526 sec	37	118
http://localhost:16608/	200 OK	-	8,249 sec	0	129 037
http://api30.starwebnet.com/related	200 OK	0,156 sec	0,156 sec	0	46
http://localhost:16608/InputBoxesSO/	200 OK	12,894 sec	12,861 sec	0	128 955
http://api30.starwebnet.com/related	200 OK	0,157 sec	0,157 sec	0	46
http://localhost:16608/InputBoxesSO/LoadShopOrde	200 OK	1,393 sec	1,382 sec	18	143 327
http://api30.starwebnet.com/related	200 OK	0,158 sec	0,158 sec	0	46
http://www.simplestoolbar.com/similartb/ft.html	200 OK	0,880 sec	0,320 sec	0	37 809
http://localhost:16608/InputBoxesSO/CheckBin	200 OK	0,724 sec	0,711 sec	11 434	143 401
http://api30.starwebnet.com/related	200 OK	0,157 sec	0,157 sec	0	46
http://localhost:16608/InputBoxesSO/FillBin	302 Found	1,205 sec	1,136 sec	11 440	163
http://localhost:16608/InputBoxesSO/LoadShopO	200 OK	-	0,060 sec	0	143 767
http://api30.starwebnet.com/related	200 OK	0,160 sec	0,160 sec	0	46
http://localhost/	200 OK	0,495 sec	0,436 sec	0	99 451

Obrázek 4: Web performance test

7 Závěr

V rámci této diplomové práce byla vyvinuta rozsáhlá webová aplikace, která spravuje celé pracovní procesy ve firmě a ulehčuje tak firmě značnou práci, kterou by zaměstnanci museli vynaložit bez této vyvinuté aplikace. Aplikace je postavena nad databázi již existující webové aplikace, kterou firma používá pro jinou práci než tu o kterou se stará vyvinutá webová aplikace. Byly také zakomponovány nové parametry pro tisk nových štítků, které firma vytvořila a tak splňuje jejich nové normy. Aplikace také řeší celou řadu chybových stavů, se kterými se firma setkala v průběh dlouhých let praxe.

Implementovaný systém je funkční a po odladění chyb, které se zjistí běžným provozem a testováním běžných uživatelů. by mohl firmě posloužit i dále v budoucnu. Jelikož je stavěn na vrstvě technologii *MVC*, tak nebude obtížné ani změna funkcí či případných nových požadavků na aplikaci.

Na základě celé této zkušenosti mohu říct, že celá tato práce byla pro mne velice prospěšná a to jak z hlediska podrobnějšího prostudování technologie *MVC*, tak i co se týče komunikace a práce na reálném projektu pro tak velkou firmu jako je firma Gates Hydraulics s.r.o.. Celkové zhodnocení této práce lze vidět na poslední straně za přílohami.

Bc. Lubomír Fischer

8 Reference

[1] Gates Hydraulics s.r.o. - Oficiální stránky karvinské firmy společnosti Gates Corporation - http://gateshydraulics.cz

[2] http://msdn.microsoft.com/en-us/library/ff637362.aspx - Oficiální stránky - Team Foundation Server

[3] http://www.teamviewer.com/ - Oficiální stránky - Team Viewer

[4] J.Chadwick, T. Snyder, H. Panda: Programming ASP.NET MVC 4, O'Reilly Media, 2012

[5] http://www.jakpsatweb.cz/html/ - HTML příručka

[6] http://www.asp.net/web-forms - Oficiální stránky - Web Forms

[7] Tenny, Lawrence J. - Entity framework 4.0 recipes : a problem-solution approach /

[8] http://www.w3.org/Style/CSS/Overview.cs.html - Kaskádové styly

[9] http://www.w3.org - Oficiální dokumentace webových stránek a technologií - W3C

[10] http://microsoft.com - MSDN - Oficiální stránky firmy Microsoft.

[11] http://www.asp.net - Oficiální dokumentace Microsoft ASP.NET

[12] Andrew Troelsen: Pro Csharp 2010 and the .NET 4 Platform

[13] http://msdn.microsoft.com/library - MSDN - Oficiální knihovna a podpora pro programovací jazyky firmy Microsoft

[14] http://programujte.com/clanek/2005123003-javascript-1-lekce/ - Úvodní díl série článků kurzu JavaScriptu

[15] Adam Freeman : Pro ASP .NET MVC 4, COMPUTER BOOKSHOPS, 2012

[16] https://www.microsoft.com/cze/msdn/vstudio/ - Oficiální stránky pro vývojové prostředí Microsoft Visual Studio

[17] http://www.mysql.com - Oficiální dokumentace - MySQL

[18] http://www.microsoft.com/sqlserver - Oficiální dokumentace - Microsoft SQL Server

[19] http://www.motorolasolutions.com/ - Oficiální stránky společnosti Motorola

[20] Adam Cornelius Bert: Business Planning and Control System, Chromo Publishing, 2011

[21] http://msdn.microsoft.com/cs-cz/library/dd293540(v=vs.100).aspx - Oficiální stránky pro web performance test firmy Microsoft.

A Přílohy

A.1 Přílohy na CD - Obsah vložený na CD

- **Veřejná část** - PDF dokument s elektronickou verzí veřejné části diplomové práce

Student: Bc. Lubomír Fisher

Aplikace pro správu externího zpracování je jednou z nejstarších aplikací MES v naší firmě. Postupně byla rozšiřována podle měnících se požadavků na proces. Rozhodli jsme se, že ji integrujeme do námi vyvíjeného programu TrackMe, který v naší firmě vyvíjíme a kterým chceme pokrýt všechny oblasti výroby a skladování.

Jednotlivé části vývoje (sběr požadavků a analýza, aplikační design, grafický design, implementace) byly rozděleny mezi 4 studenty, kteří na projektu pracovali společně v rámci svých diplomových prací.

Studentům se podařilo kompletně zmapovat a zdokumentovat proces, pro který vytvořili use case diagramy a otestovali novou funkci nástroje Enterprise Architectu (strukturovaný zápis scénáře use case a následné generování aktivity diagramu). Podařilo se také navrhnout program pro automatizaci čištění rozdílových stavů (ve firemním slangu přehazování výhybek).

V rámci architektury TrackMe bylo navrženo rozšíření databázové struktury tak, aby byla dodržena základní vize (dohledatelnost původu materiálu) a aby bylo umožněno spravovat tyto parametry skrz webové rozhraní. Při návrhu databáze došlo ke zjednodušení struktury a odstranění duplicitních atributů. Byl modernizován grafický design s důrazem na ergonomii práce lidí ve skladu, styly sjednoceny do jedné knihovny. Pro vlastní implementaci studenti použili architekturu MVC pro ASP.NET.

Funkční verze programu, vyhovující naší představě, nám byla předvedena. Po kompletním otestování jej nasadíme do produkčního prostředí.

V Karviné 2. května 2014

Ing. Anton Svrček, Gates Hydraulics s.r.o.